03

Pila de ejecución

Transcripción

[00:00] ¿Qué tal? Sean bienvenidos a la nueva parte de su curso de Java. Esta es la cuarta parte en la cual vamos a ver justo lo que es el tratamiento de errores o más específicamente dicho, excepciones. Para entender esto vamos a hablar un poco de lo que es la pila de ejecución, pero antes vamos a revisar nuestro setup que ya hemos configurado hasta ahora a lo largo del curso.

[00:22] Para esto estoy ahora en mi terminal, en mi consola. Yo voy a dar un Java -version para obtener la versión que tengo de Java ahora y tengo la versión 11 instalada, entonces ya tengo ya la JDK, la JDM ya configurados aquí en este computador y ahora también tengo mi IDE Eclipse que es el IDE que hemos estado utilizando para los ejemplos y los trabajos que hemos venido haciendo hasta ahora.

[00:46] Aquí está el proyecto que hemos hecho al final, el que el proyecto acerca de herencia y polimorfismo con nuestro Byte Bank. No vamos a trabajar mucho con él aquí en esta primera parte, entonces yo creo que no es necesario que lo descarguen por el momento. Y bueno, vamos a comenzar aquí creando un nuevo proyecto de Java al cual le vamos a poner Java-pila-ejecución porque es el tema que vamos a ver ahora.

[01:15] No se preocupen, le damos finish. Le damos aquí a don't create y perfecto, acá tenemos nuestra pila de ejecución. Abrimos esto y tenemos la carpeta src que bueno, no tiene nada, tiene un paquete por defecto, entonces ¿qué vamos a hacer? Vamos a crear una nueva clase. Y a esa clase la vamos a llamar flujo. ¿Por qué? Vamos a mostrar cómo es el flujo de la pila de ejecuci

[01:44] Pila de ejecución podemos hacer referencia que es el lugar en la memoria, digamos donde tienen lugar todos los eventos que Java va ejecutando, pero vamos a ver eso con más detalle más adelante. Ahora lo que yo voy a hacer aquí es copiar un código que ustedes ya deben tener en la parte de abajo, en la descripción del video.

```
public class Flujo {

  public static void main(String[] args) {
    System.out.println("Ini do main");
    metodo1();
    System.out.println("Fim do main");
  }
  public static void metodo1(){
    System.out.println("Ini do metodo1");
    metodo2();
    System.out.println("Fim do metodo1");
  }
  public static void metodo2(){
    System.out.println("Ini do metodo2");
    for(int i =1; i<= 5; i++){
      System.out.println("i");
    }
    System.out.println("Fim do metodo2");
  }
}
```

COPIA EL CÓDIGO

[02:05] Ese código es básicamente es algo bien simple, es un conjunto de métodos, método main, es ese de aquí, es un método main que es el que ya conocemos todos. Aquí tenemos nuestro inicio de main y aquí tenemos el fin

de main. Perfecto. Vemos que este método main llama aquí dentro a un método 1. ¿Y quién es método 1? Este método que está aquí abajo.

[02:36] Vemos aquí que Eclipse ya nos ayuda, pues resaltando, si yo selecciono aquí método 1, él automáticamente me sombrea dónde está ubicado método 1. Y aquí de igual forma, inicio método 1 y acá es fin de método 1. Perfecto. Pero aquí dentro vemos que también él está llamando a método 2. ¿Quién es método 2? Él está aquí abajo. Y misma estrategia.

[03:05] Inicio método 2. Fin método 2. Pero en método 2, él está haciendo una iteración, un for, él está imprimiendo básicamente los números del 1 hasta el 5, porque vemos que el índice comienza en 1 y va a parar cuando sea igual o mayor a 5. Vemos que todos son estáticos y por qué son estáticos, porque si estamos llamándolos directamente en el método main, él precisa ser estático, porque si no, él va a dar un error de compilación aquí.

[03:36] ¿Qué pasaría si yo no lo quiero llamar de forma estática? Tendría que inicializar mi objeto flujo dentro del método main y ahí llamar al método 1, que pertenece al método, a la clase flujo. Eso ya lo hemos visto en los cursos anteriores. Por el momento lo vamos a dejar en estático porque no necesitamos hacer mucho ahí. Y bueno, ya hemos visto un pequeño overview así bien alto nivel de qué está haciendo este código método 1, método 2, qué es lo que ellos deberían hacer.

[04:09] Entonces, ahora vamos a ejecutarlo, guardamos. Y le vamos a dar al signo play como siempre para ejecutarlo, y tenemos aquí esto. Inicio de main, inicio método 1. Y después él va a inicio método 2, o sea, él imprime esta línea de aquí, llega a método 1, automáticamente él baja a método 1, imprime Inicio de método 1. Nuevamente aquí tiene método 2.

[04:43] ¿Él qué hace? Entra a método 2 e imprime Inicio del método dos. Él hace este for de aquí y él va a imprimir los números del 1 al 5. Y después, ¿qué va a hacer? Fin del método 2 nuevamente y él regresa a método 1. Miren qué

curioso, él regresa a método 1, imprime Fin de método 1 y, a la vez, él regresa a main y él dice Fin main.

[05:14] Entonces esto que hemos visto ahora es lo que llevamos la pila de ejecución. Esto no es digamos una orientación a objetos. Esto de hecho, es una programación bien, bien procedural, como ya hemos visto, lenguajes como C, C++, que de hecho son los lenguajes donde Java está basado. Java está basado en C. Incluso el nombre del método main es el nombre que usa el método C como método principal para inicializar un programa.

[05:45] Entonces hasta ahora no hemos visto nada nuevo pero sí hemos visto en acción la pila de ejecución. En el próximo video vamos a ver mucho más a fondo, más explicado, cómo es que funciona esto de la pila de ejecución y para qué se usa. Nos vemos.

Hagas clic aquí (https://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Exception.ht para acceder a la documentación oficial y obtener más información sobre la clase Exception.